Stat 243

Base Graphics (part 1)

Gaston Sanchez

Graphics

References

Some Resources

- **R Graphics** by Paul Murrell
- **R Graphics Cookbook** by Winston Chang
- **ggplot2: Elegant Graphics for Data Analysis** by Hadley Wickham
- **R Graphs Cookbook** by Hrishi Mittal
- **Graphics for Statistics and Data Analysis with R** by Kevin Keen

2 main graphics systems

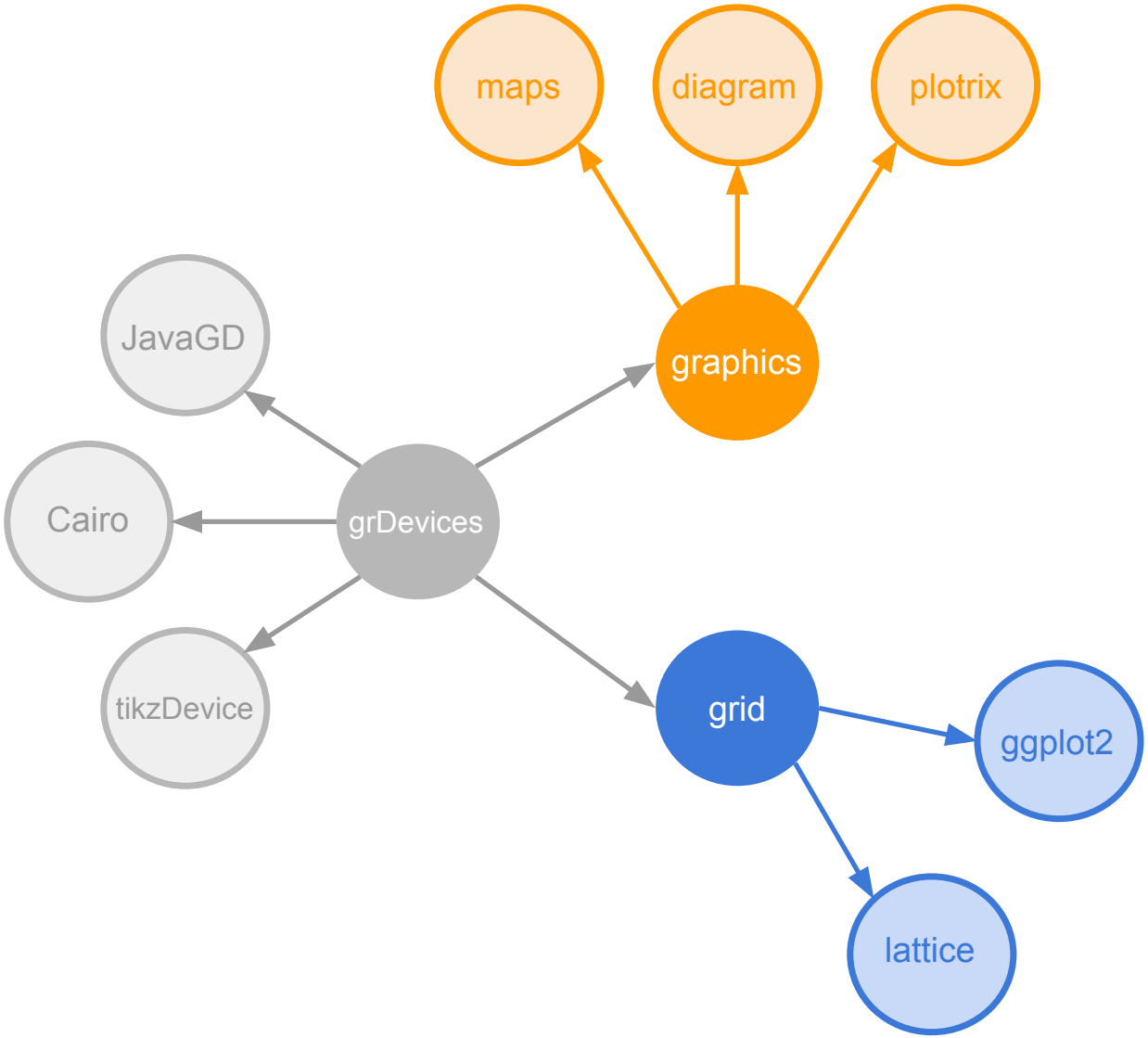
"graphics" & "grid"

Graphics Systems

- `"graphics"` and `"grid"` are the two main graphics systems in R
- `"graphics"` is the *traditional* system, also referred to as *base graphics*
- `"grid"` prodives low-level functions for programming plotting functions

Graphics Engine

- Underneath `"graphics"` and `"grid"` there is the package `"grDevices"`
- `"grDevices"` is the graphics **engine** in R
- It provides the graphics devices and support for colors and fonts


maps
diagram
plotrix
JavaGD
graphics
Cairo
grDevices
tikzDevice
grid
ggplot2
lattice

Package "graphics"

The package "graphics" is the traditional system; it provides functions for complete plots, as well as low-level facilities.

Many other graphics packages are built on top of graphics like "maps", "diagram", "pixmap", and many more.

Package "grid"

The "grid" package does not provide functions for drawing complete plots.

"grid" is not used directly to produce statistical plots. Instead, it is used to build other graphics packages like "lattice" or "ggplot2".

Traditional (Base) Graphics

Types of graphics functions

Graphics functions can be divided into two main types:

- **high-level** functions produce complete plots, e.g. `barplot()`, `boxplot()`, `dotchart()`
- **low-level** functions add further output to an existing plot, e.g. `text()`, `points()`, `legend()`

The `plot()` function

- `plot()` is the most important high-level function in traditional graphics
- The first argument to `plot()` provides the data to plot
- The provided data can take different forms: e.g. vectors, factors, matrices, data frames.
- To be more precise, `plot()` is a generic function
- You can create your own `plot()` method function

Basic Plots with `plot()`

In its basic form, we can use `plot()` to make graphics of:

- one single variable
- two variables
- multiple variables

Plots of One Variable

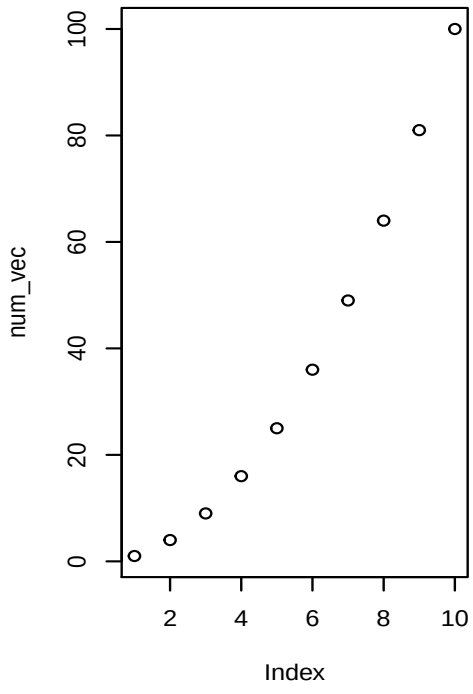
High-level graphics of a single variable

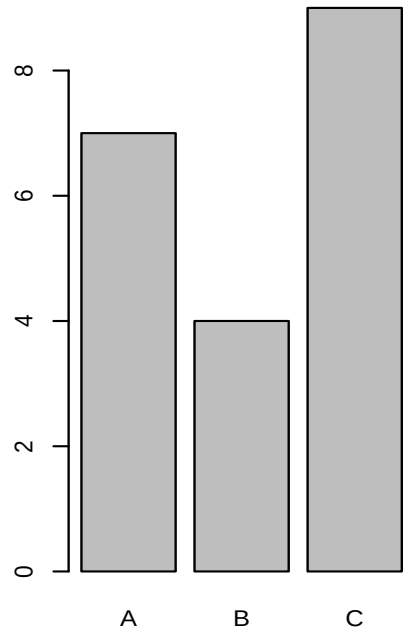
Function	Data	Description
`plot()`	numeric	scatterplot
`plot()`	factor	barplot
`plot()`	1-D table	barplot

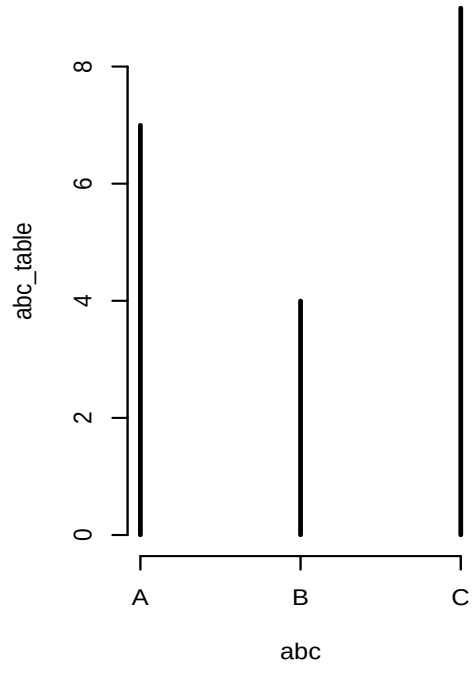
`numeric` can be either a vector or a 1-D array (e.g. row or column from a matrix)

Examples

```r
# plot numeric vector
num_vec <- (c(1:10))^2
plot(num_vec)

# plot factor
set.seed(4)
abc <- factor(sample(c('A', 'B', 'C'), 20,
                     replace = TRUE))
plot(abc)

# plot 1D-table
abc_table <- table(abc)
par(op)
```


num_vec
0
20
40
60
80
100
2
4
6
8
10
Index


0
2
4
6
8
A
B
C


abc_table
0
2
4
6
8
A
B
C
abc

More high-level graphics of a single variable

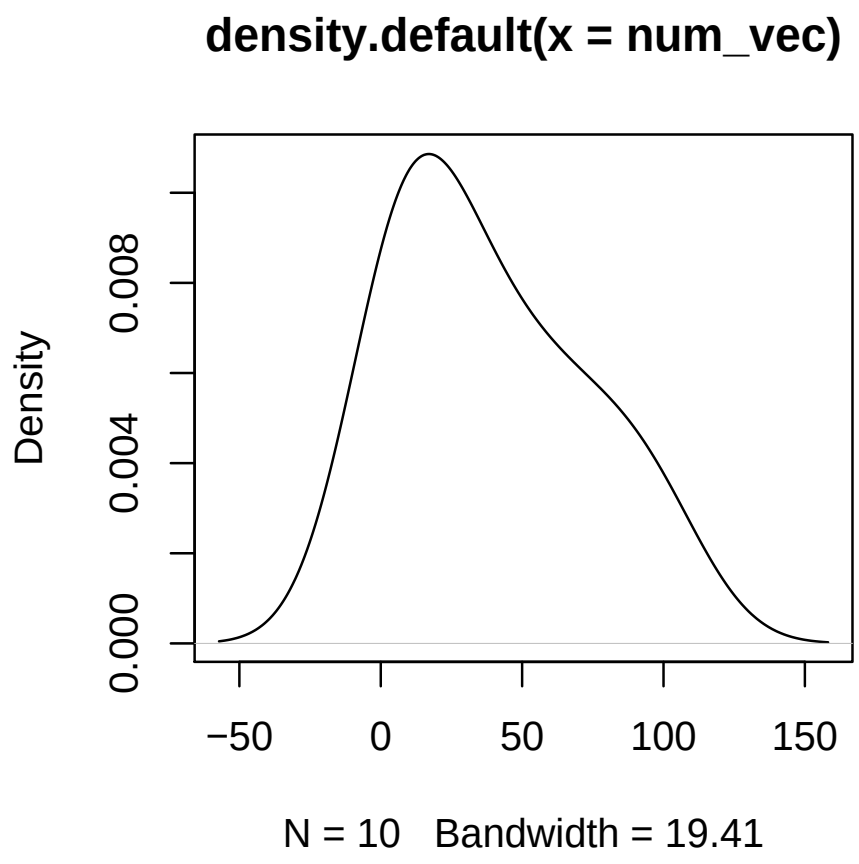
Function	Data	Description
`barplot()`	numeric	barplot
`pie()`	numeric	pie chart
`dotchart()`	numeric	dotplot
`boxplot()`	numeric	boxplot
`hist()`	numeric	histogram
`stripchart()`	numeric	1-D scatterplot
`stem()`	numeric	stem-and-leaf plot

Kernel Density Curve

- Surprisingly, R does not have a specific function to plot density curves
- R does have the `density()` function which computes a kernel density estimate
- We can pass a `"density"` object to `plot()` in order to get a density curve.

Kernel Density Curve

```
# kernel density curve
dens <- density(num_vec)
plot(dens)
```



Plots of Two Variables

High-level graphics of two variables

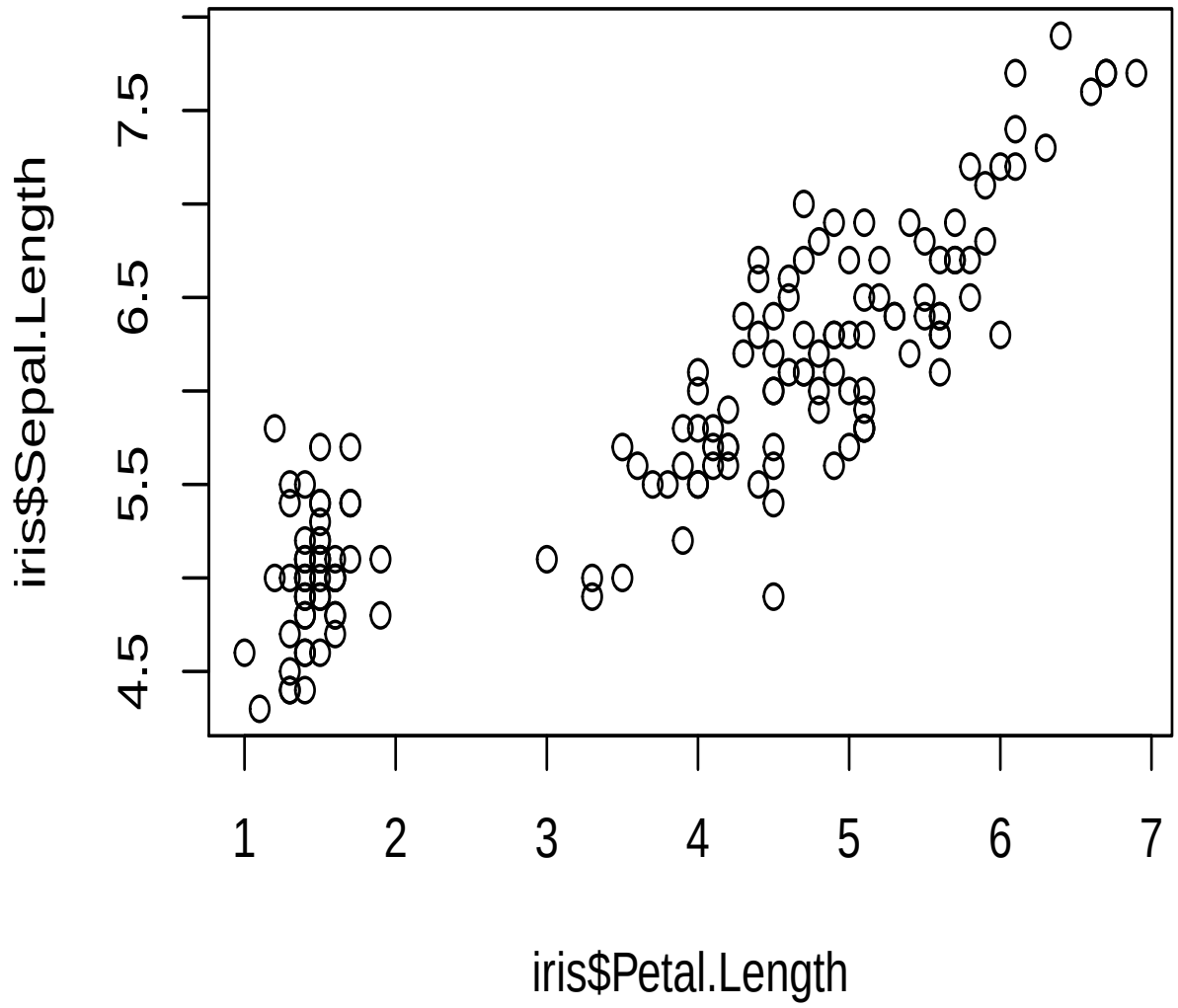
Function	Data	Description
`plot()`	numeric, numeric	scatterplot
`plot()`	numeric, factor	stripcharts
`plot()`	factor, numeric	boxplots
`plot()`	factor, factor	spineplot
`plot()`	2-column numeric matrix	scatterplot
`plot()`	2-column numeric data.frame	scatterplot
`plot()`	2-D table	mosaicplot

Plots of two variables

```r
# plot numeric, numeric
plot(iris$Petal.Length, iris$Sepal.Length)

# plot numeric, factor
plot(iris$Petal.Length, iris$Species)

# plot factor, numeric
plot(iris$Species, iris$Petal.Length)

# plot factor, factor
plot(iris$Species, iris$Species)
```

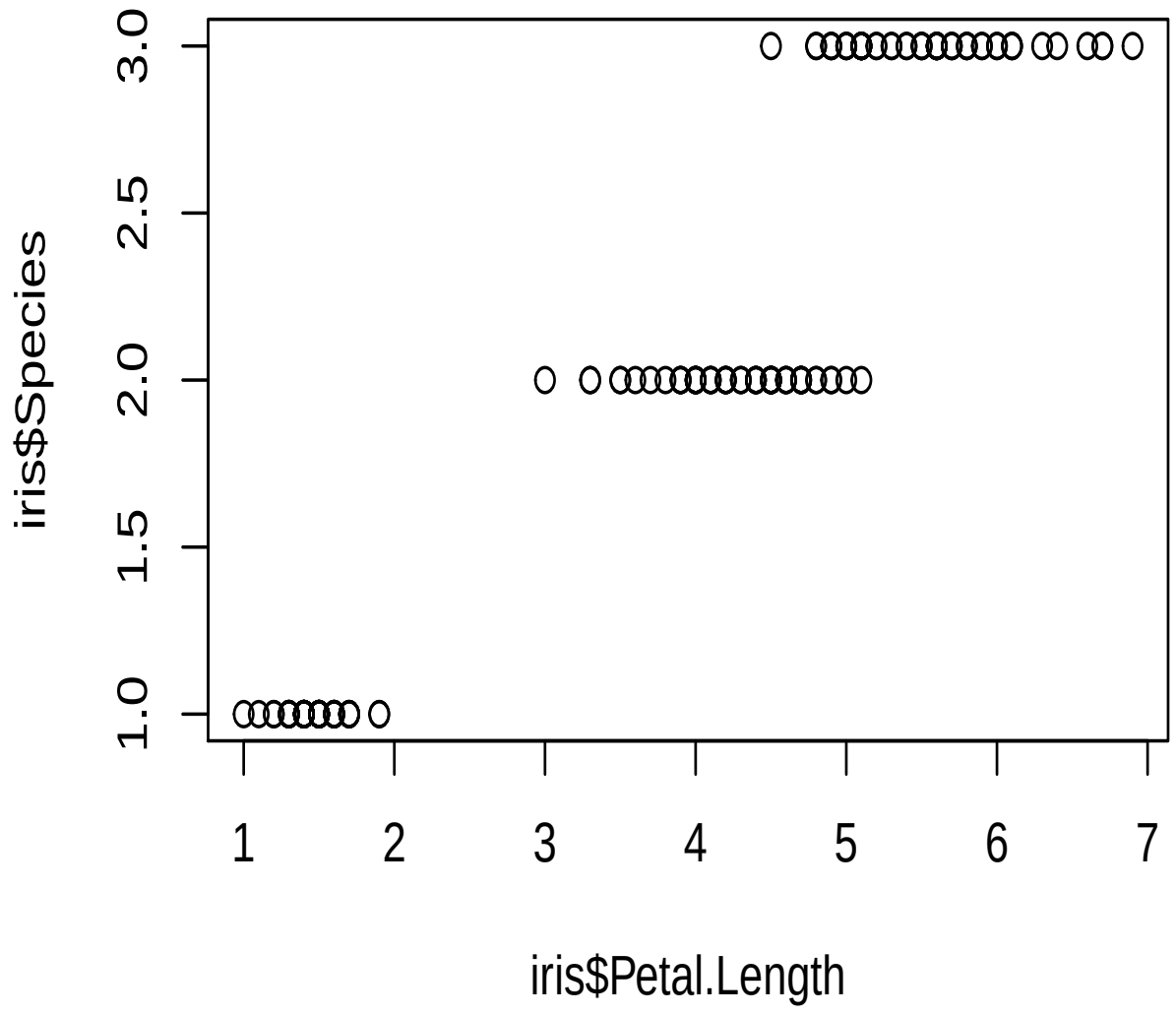
Plots of two variables

```
# plot numeric, numeric
plot(iris$Petal.Length, iris$Sepal.Length)
```



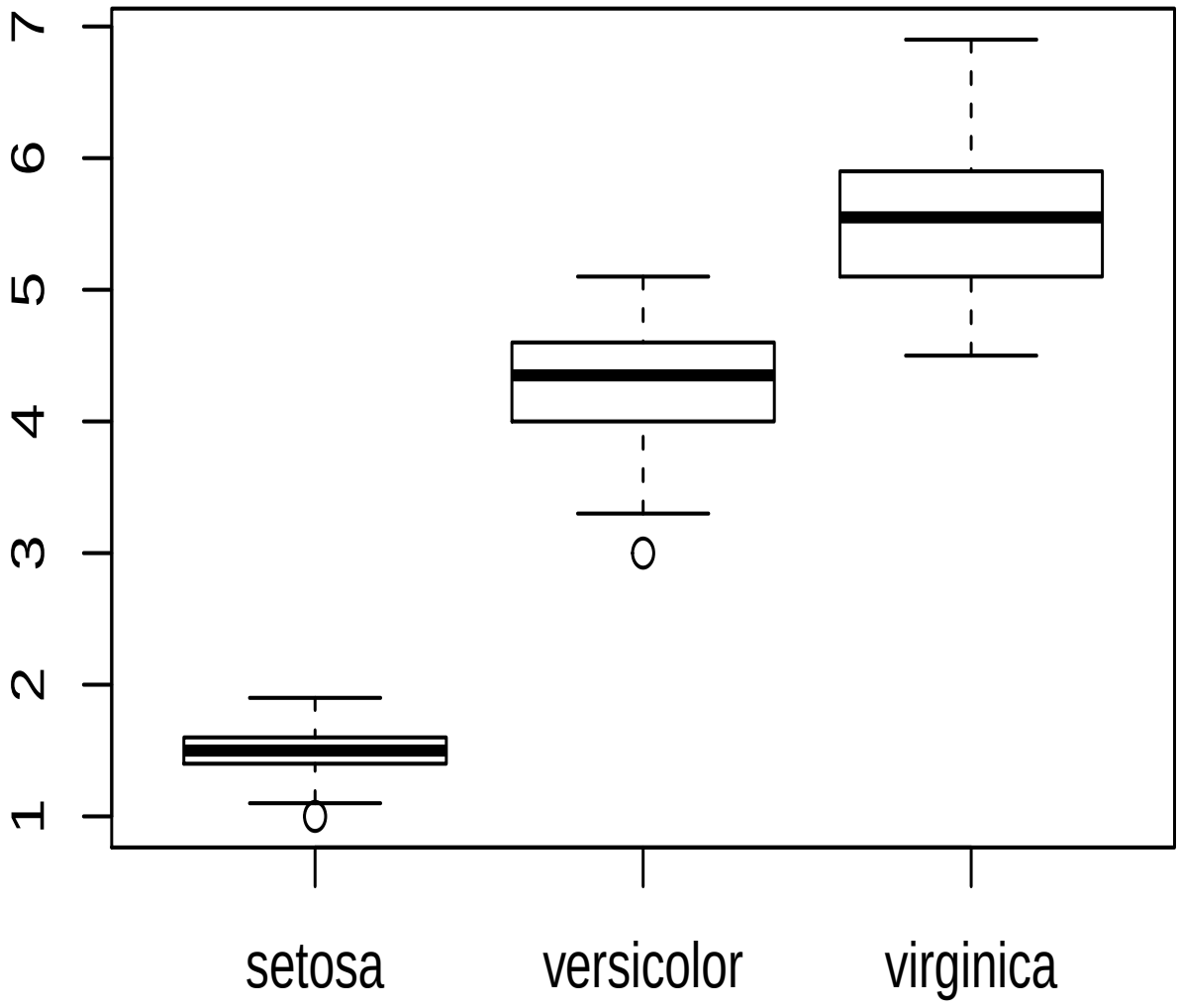
Plots of two variables

```
# plot numeric, factor
plot(iris$Petal.Length, iris$Species)
```



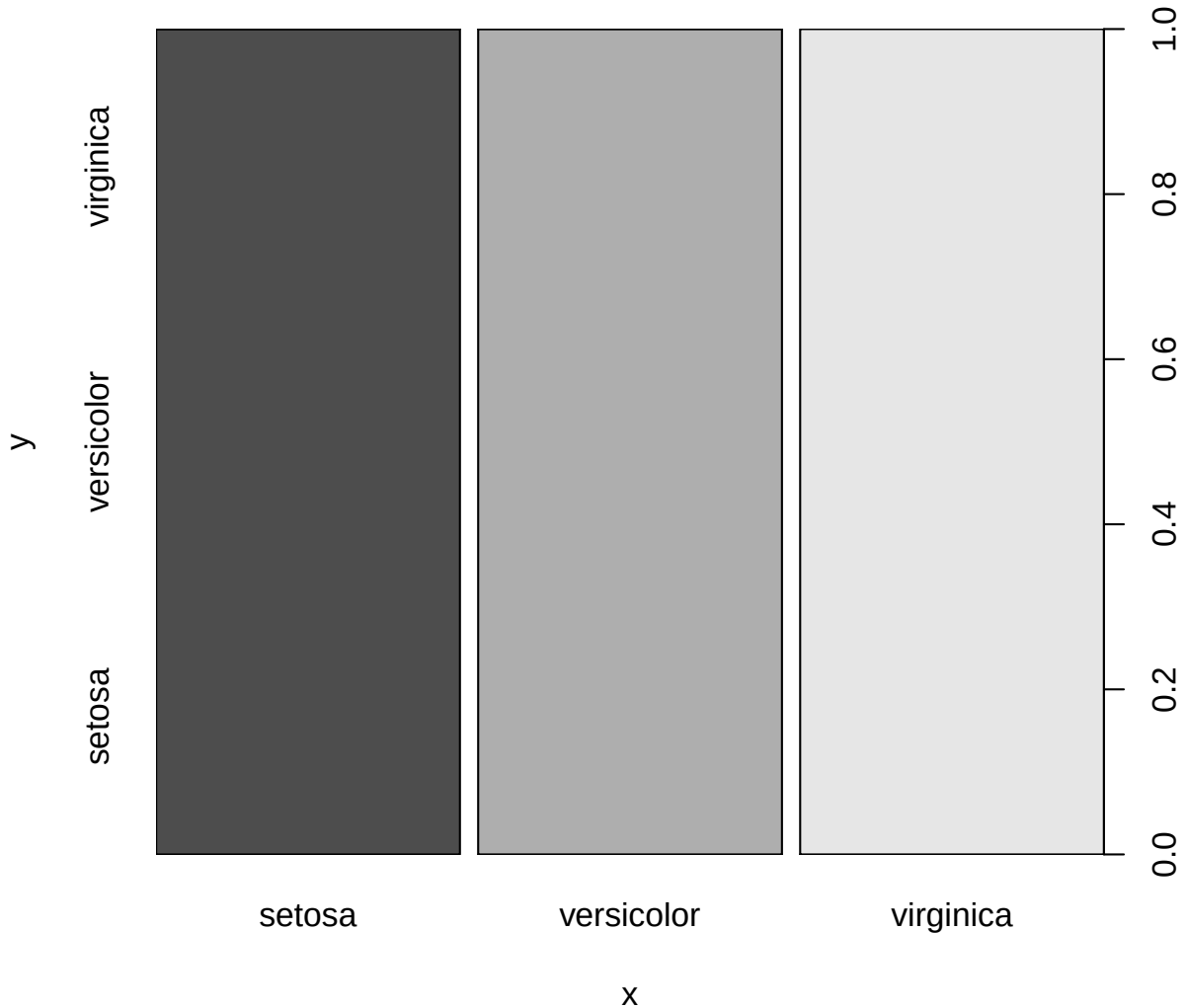
Plots of two variables

```
# plot factor, numeric
plot(iris$Species, iris$Petal.Length)
```



Plots of two variables

```
# plot factor, factor
plot(iris$Species, iris$Species)
```



More high-level graphics of two variables

Function	Data	Description
`sunflowerplot()`	numeric, numeric	sunflower scatterplot
`smoothScatter()`	numeric, numeric	smooth scatterplot
`boxplot()`	list of numeric	boxplots
`barplot()`	matrix	stacked / side-by-side barplot
`dotchart()`	matrix	dotplot
`stripchart()`	list of numeric	stripcharts
`spineplot()`	numeric, factor	spinogram
`cdplot()`	numeric, factor	conditional density plot
`fourfoldplot()`	2x2 table	fourfold display
`assocplot()`	2-D table	association plot
`mosaicplot()`	2-D table	mosaic plot

Base Graphics

Graphics in R

Traditional Graphics

- R `"graphics"` follows a static, "painting on canvas" model.
- Graphics elements are drawn, and remain visible until painted over.
- For dynamic and/or interactive graphics, base R graphics are limited.

Traditional graphics model

In the traditional model, we create a plot by first calling a high-level function that creates a complete plot, and then we call low-level functions to add more output if necessary

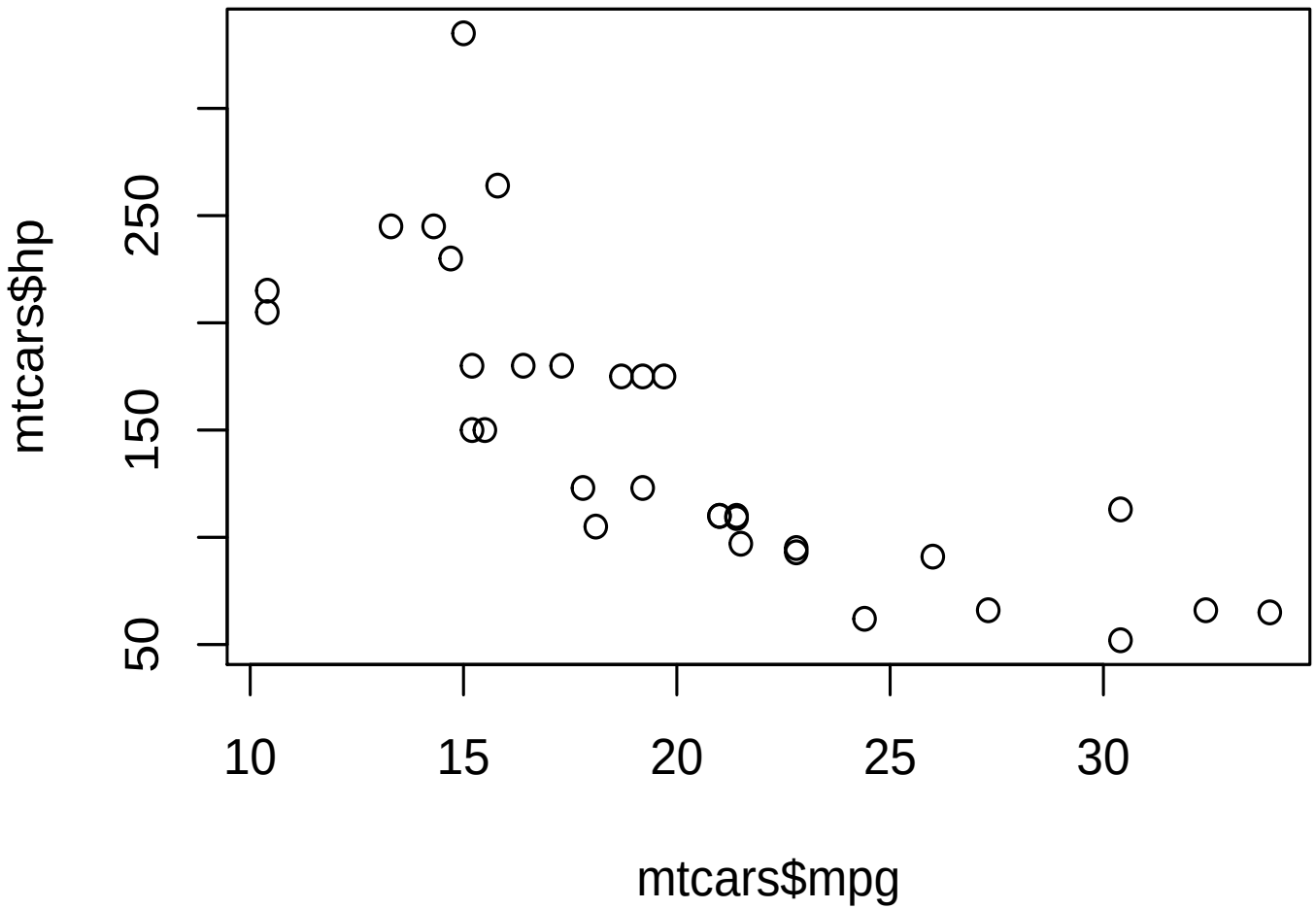
Dataset mtcars

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs
## Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0
## Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0
## Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1
## Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0
## Valiant          18.1   6  225 105 2.76 3.460 20.22  1
```

Scatter plot

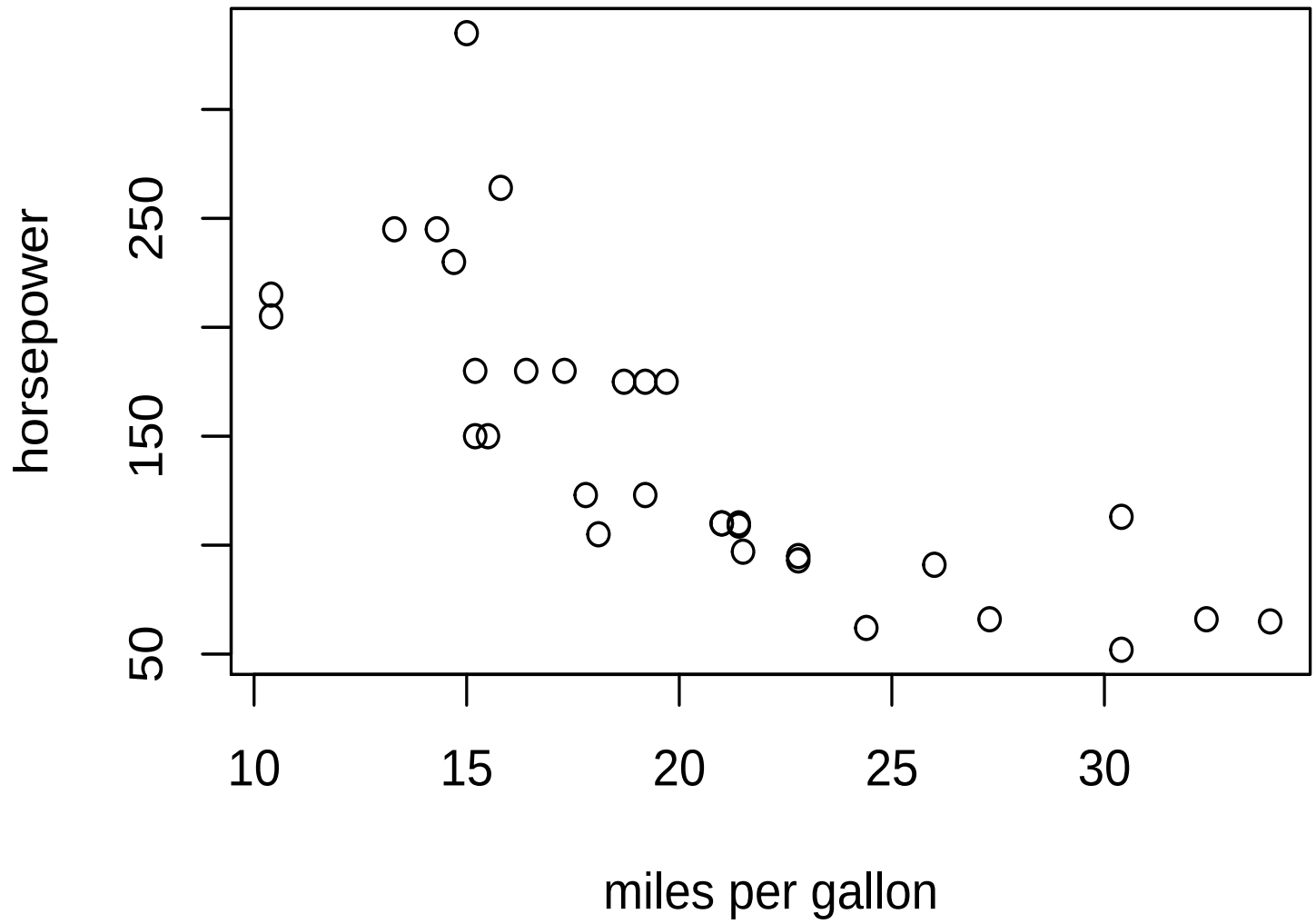
```
# simple scatter-plot
plot(mtcars$mpg, mtcars$hp)
```



Axis Labels

```r
# xlab and ylab
plot(mtcars$mpg, mtcars$hp,
     xlab = "miles per gallon",
     ylab = "horsepower")
```

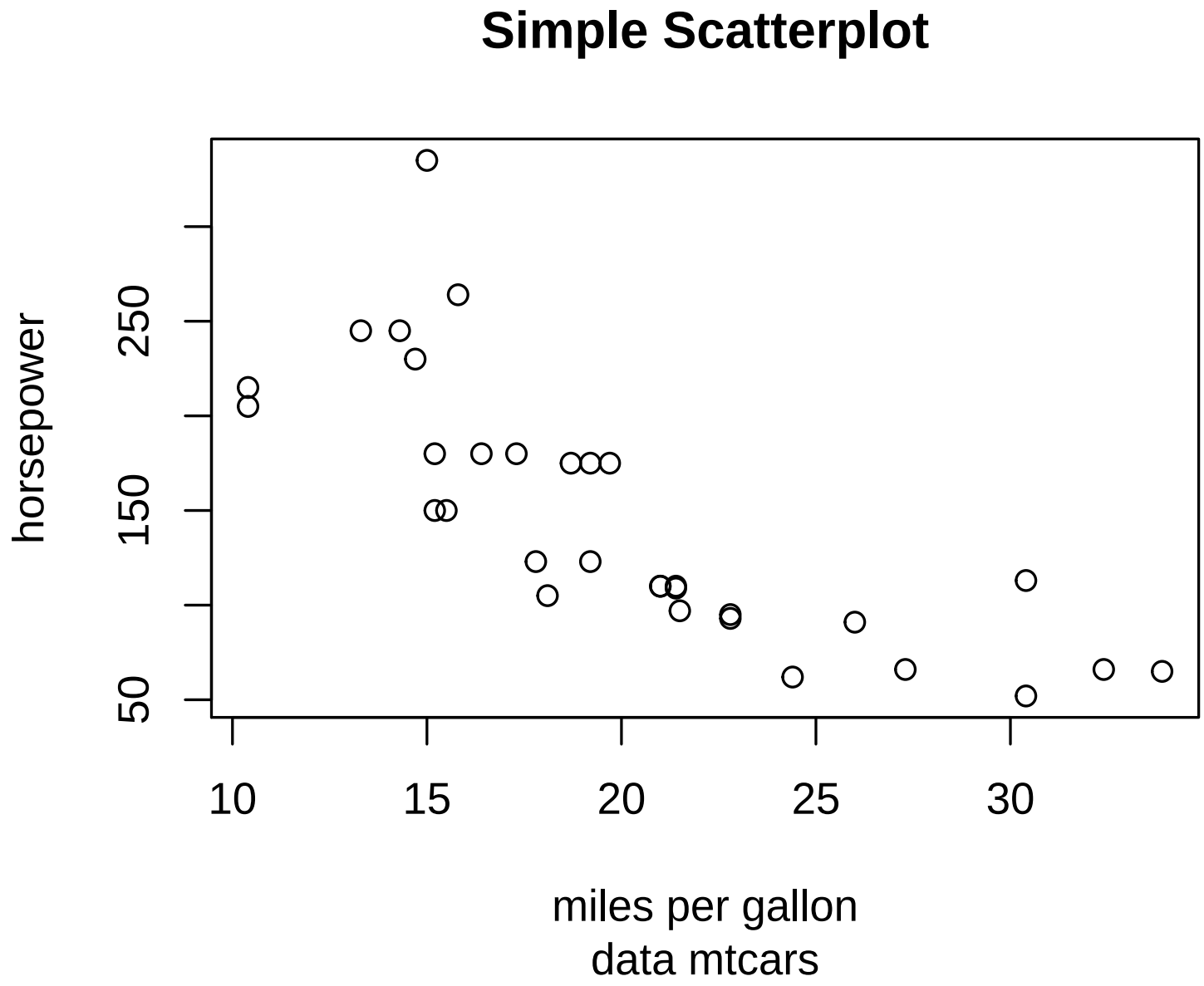
Axis Labels



Title and Subtitle

```
# title and subtitle
plot(mtcars$mpg, mtcars$hp,
     xlab = "miles per gallon",
     ylab = "horsepower",
     main = "Simple Scatterplot",
     sub = 'data mtcars')
```

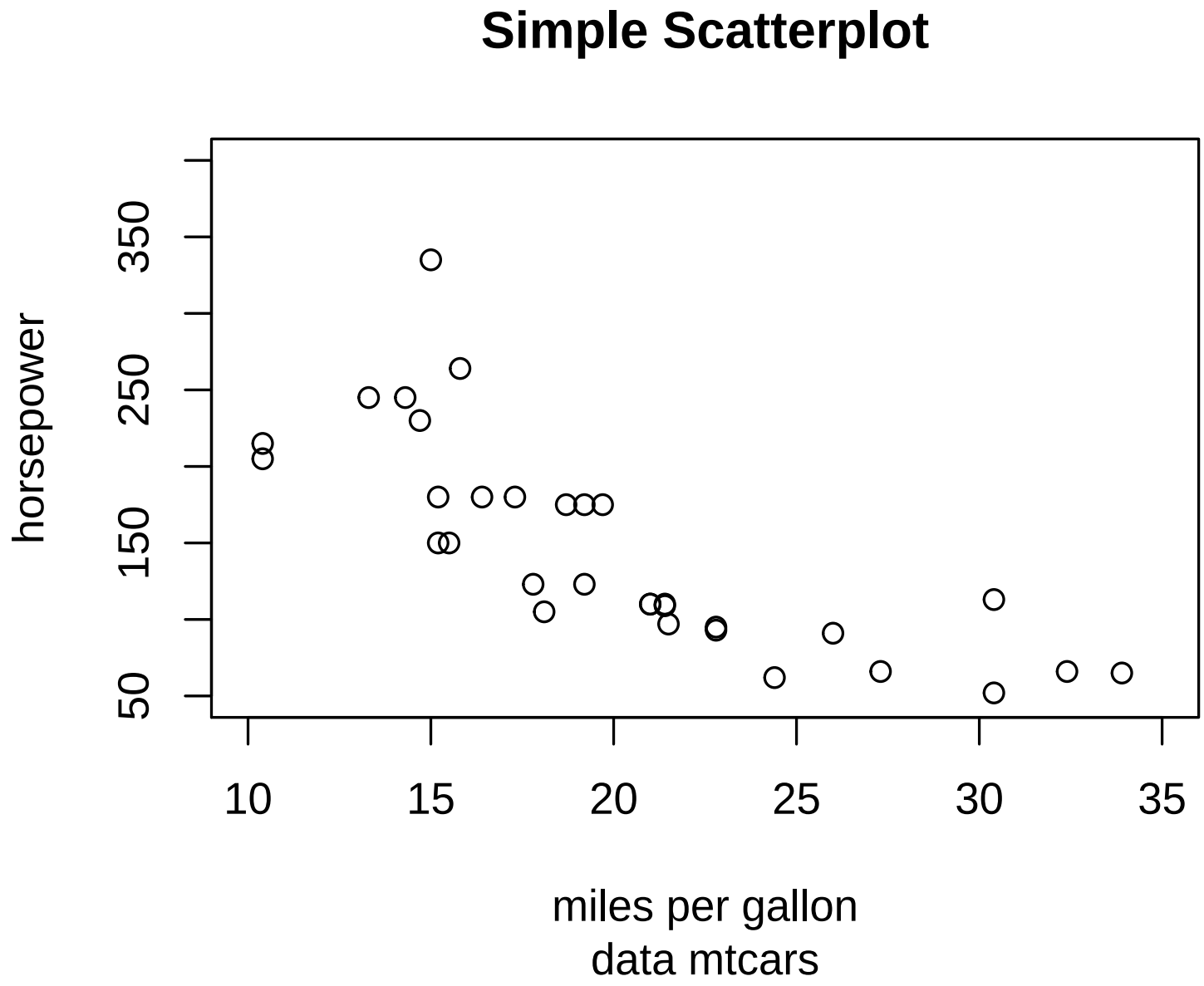
Title and Subtitle



x and y coordinate ranges

```
# 'xlim' and 'ylim'
plot(mtcars$mpg, mtcars$hp,
     xlim = c(10, 35),
     ylim = c(50, 400),
     xlab = "miles per gallon",
     ylab = "horsepower",
     main = "Simple Scatterplot",
     sub = 'data mtcars')
```

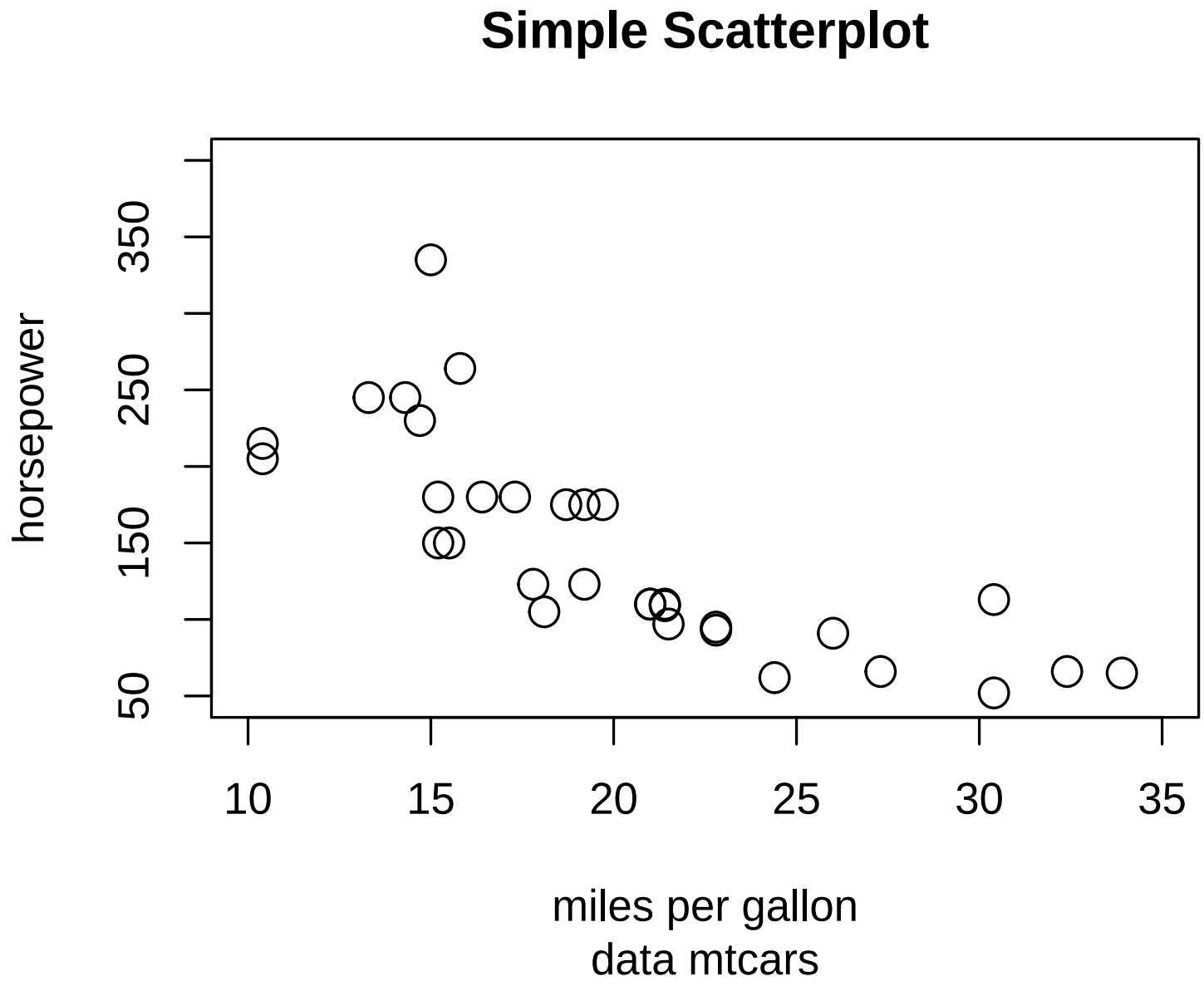
x and y coordinate ranges



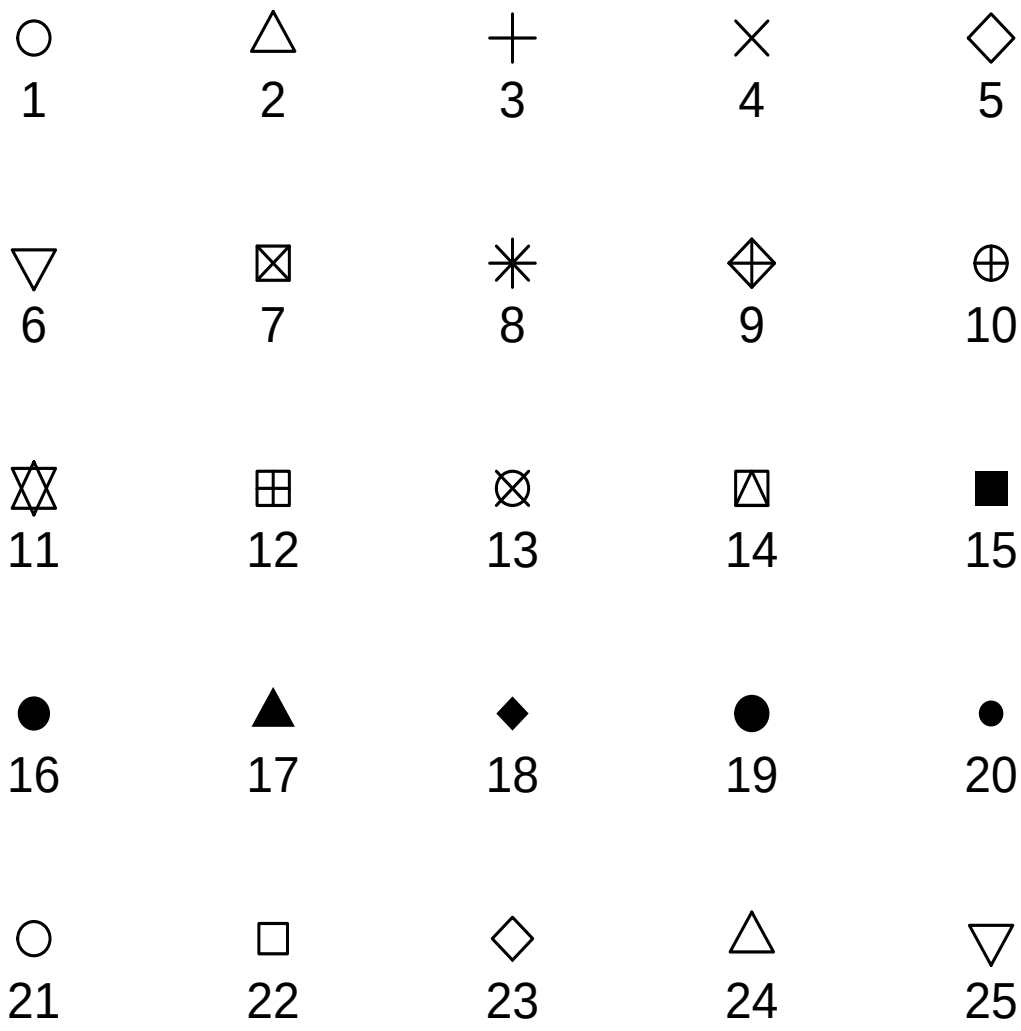
Points

```
# character expansion 'cex' and 'point character'
plot(mtcars$mpg, mtcars$hp,
     xlim = c(10, 35),
     ylim = c(50, 400),
     cex = 1.5,
     pch = 1,
     xlab = "miles per gallon",
     ylab = "horsepower",
     main = "Simple Scatterplot",
     sub = 'data mtcars')
```

Points

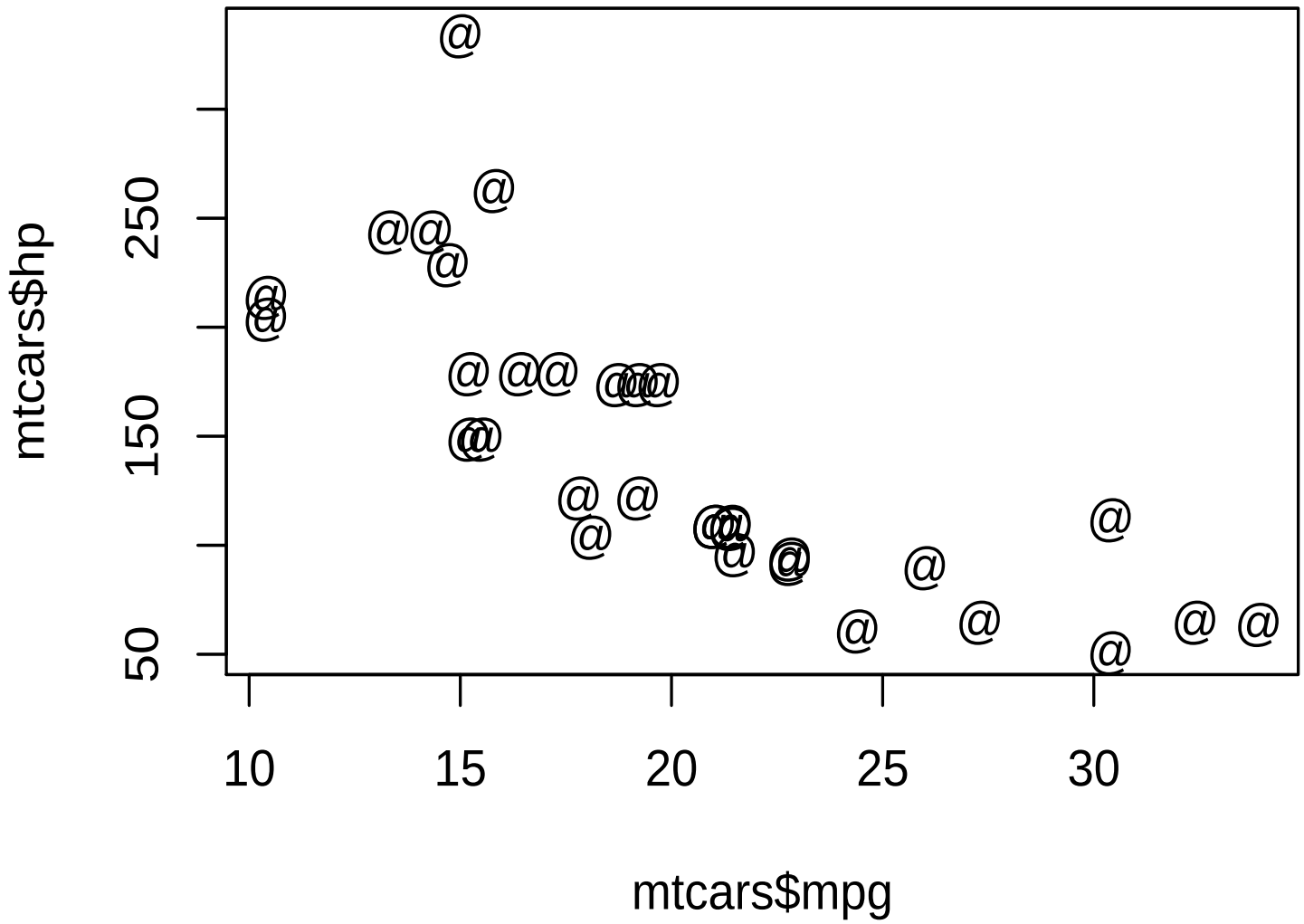

Simple Scatterplot
horsepower
50
150
250
350
10
15
20
25
30
35
miles per gallon
data mtcars

Point symbols `pch` available in R



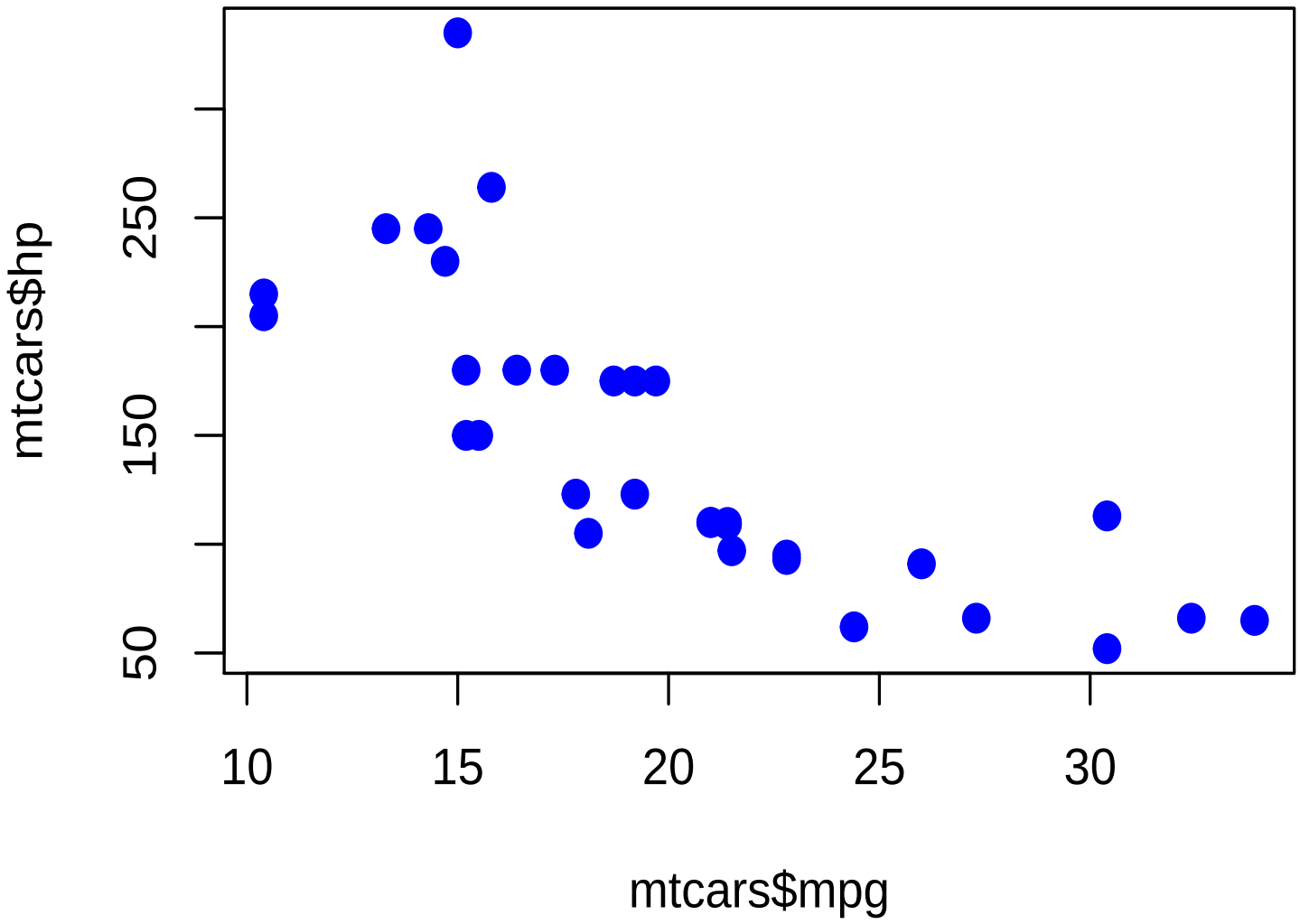
Point Character

```
# 'pch' can be any character
plot(mtcars$mpg, mtcars$hp, pch = "@")
```



Point Colors

```
# color argument 'col'
plot(mtcars$mpg, mtcars$hp, pch=19, col="blue", cex=1.2)
```



Coloring Point Symbols

- the `col` argument can be used to color symbols
- symbols 21 through 25 can additionally have their interiors filled by using the `bg` (background) argument

Coloring Point symbols

Drawing Error Bars

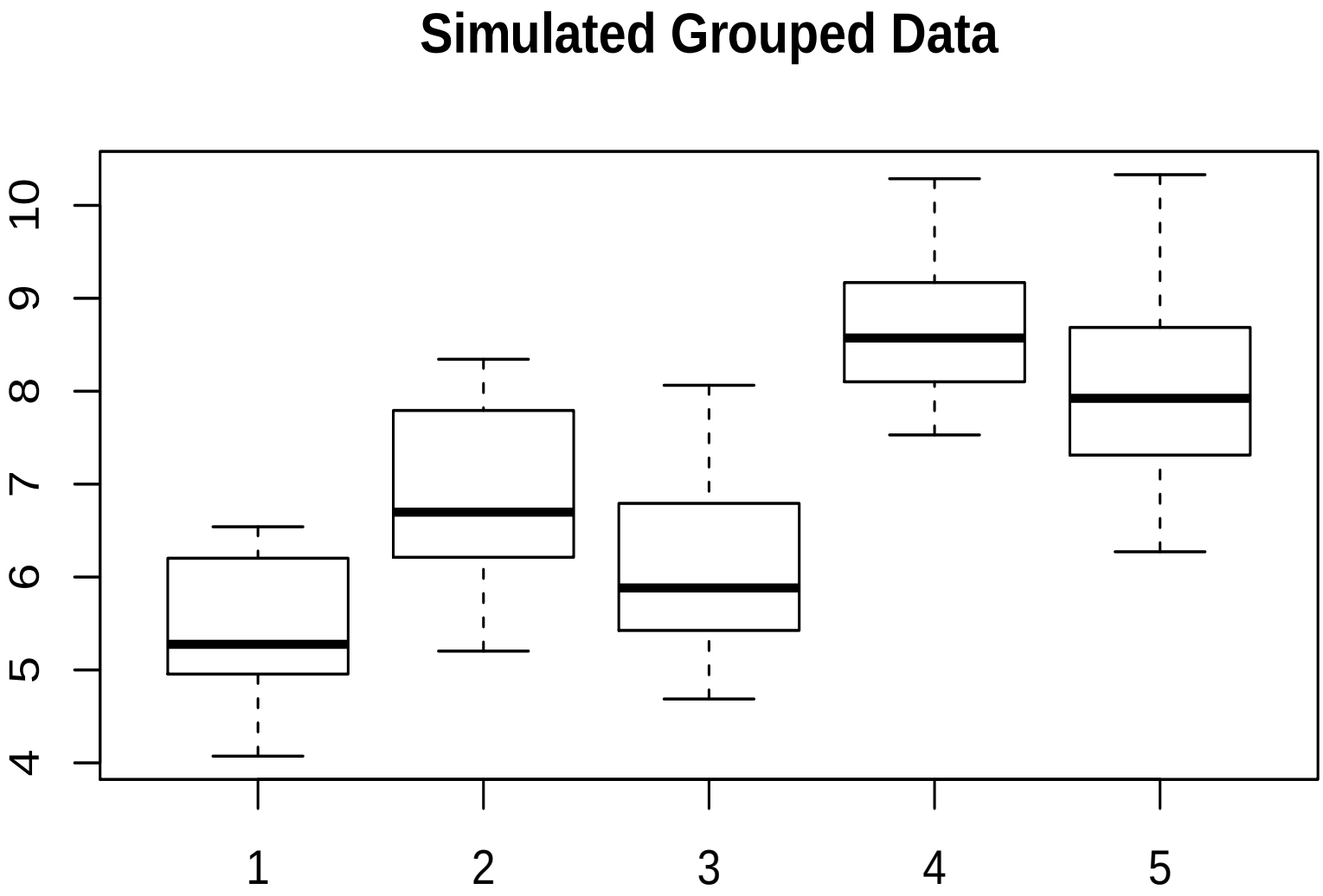
Randomly generate 5 groups of 20 observations each.

```
groups <- rep(1:5, each = 20)
means <- c(5, 7, 6, 9, 8)
y <- rnorm(100) + rep(means, each = 20)
```

Display the data is with parallel boxplots.

```
boxplot(y ~ groups, main = "Simulated Grouped Data")
```

Grouped Box-plots

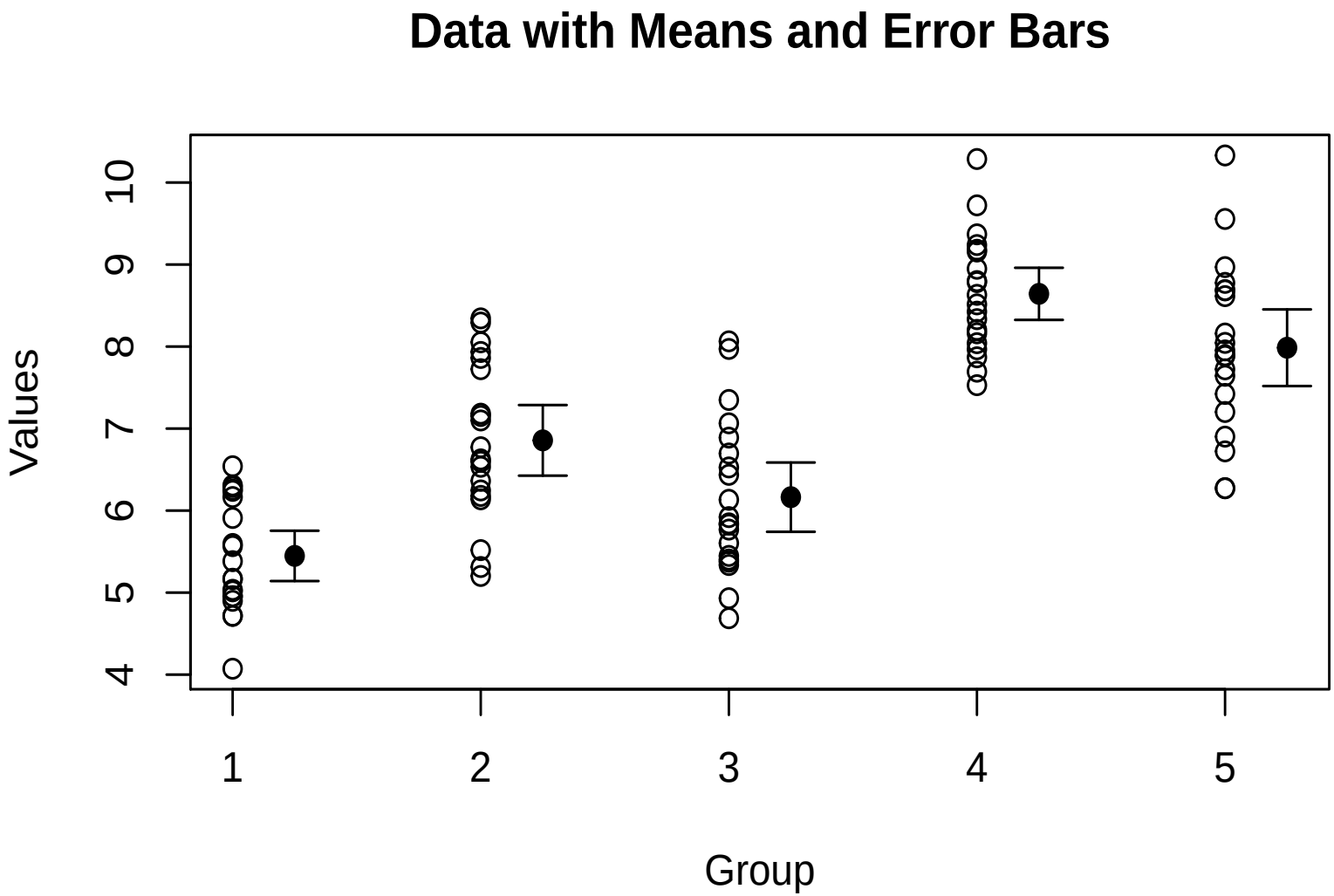


Error bars

Rather than using boxplots, we can create a new plot that displays the data values and shows the mean with error bars.

```
ybar <- sapply(split(y, groups), mean)
ysd <- sapply(split(y, groups), sd)
plot(groups, y, xlim = c(1, 5.25),
     xlab = "Group", ylab = "Values",
     main = "Data with Means and Error Bars")
points(1:5 + .25, ybar, pch=19)
arrows(1:5 + .25, ybar - 2 * ysd/sqrt(20),
       1:5 + .25, ybar + 2 * ysd/sqrt(20),
       code = 3, angle = 90, length = .1)
```

Error bars



Adding some jitter

Notice that it is hard to see the individual data points because of overplotting. This can be fixed (adding a small amount of random variation in the x direction).

```
plot(groups + runif(100, -.05, .05), y,
     xlim = c(1, 5.25),
     xlab = "Group", ylab = "Values",
     main = "Data with Means and Error Bars")
points(1:5 + .25, ybar, pch=19)
arrows(1:5 + .25, ybar - 2 * ysd/sqrt(20),
       1:5 + .25, ybar + 2 * ysd/sqrt(20),
       code = 3, angle = 90, length = .1)
```

Adding jitter